VZ-200 BASIC PROGRAM STORAGE & LINE RENUMBERING

GRAHAM MARSDEN

The VZ-200 does not have a RENUMBER command so trying to modify a program with insufficient vacant line numbers is not a welcome task. This program enables the line numbers of a program to be reset using any start number and increment providing they meet certain conditons.

In order to understand the operation of the program it is necessary to understand how a BASIC program listing and its line numbers are stored in memory.

Each line of program is formated as below:-

- The first two bytes of the sequence hold the address, in two byte form, of the first byte of the sequence for the next program line. i.e. the location holding the R above is in location P+256*Q
- The third and fourth bytes hold the line number. i.e. in this case the line number will be L+256*I
- Then the contents of the program line follow, terminating with a byte containing the value zero.

For example suppose the line:-
300 PRINT"!":GOTO400
was stored starting at address 38420. This would be the contents of locations 38420 - 38433

Note that characters (including line numbers used within a program line after GOTO or GOSUB) are stored as their ASC codes.

"Operators" like PRINT, GOTO, etc have their own single byte codes which represent the operation. The program looks for the codes for GOTO and GOSUB (amongst others- see explanation of program operation) in order to find the locations of line numbers within program lines. The codes for various operations can be determined by putting in a line.

using the operation in question, (ensure it has the lowest of all line numbers) and then type in -

FORZ = 31469 TO 31469+N:PRINT PEEK(Z);:NEXTZ
where N is the number of memory positions that you wish to see codes for. 31469 is the position in memory of the first item of the first BASIC program line. i.e. the one immediately after the line number bytes. The BASIC Program listing normally starts at 31465 unless moved - but that is another story.

Having understood how a BASIC program is stored it is possible to make changes to it without having to edit it on screen.

One thing that can be done is to change all the line numbers so they follow a constant incremant.

Here is a program to do just that:-

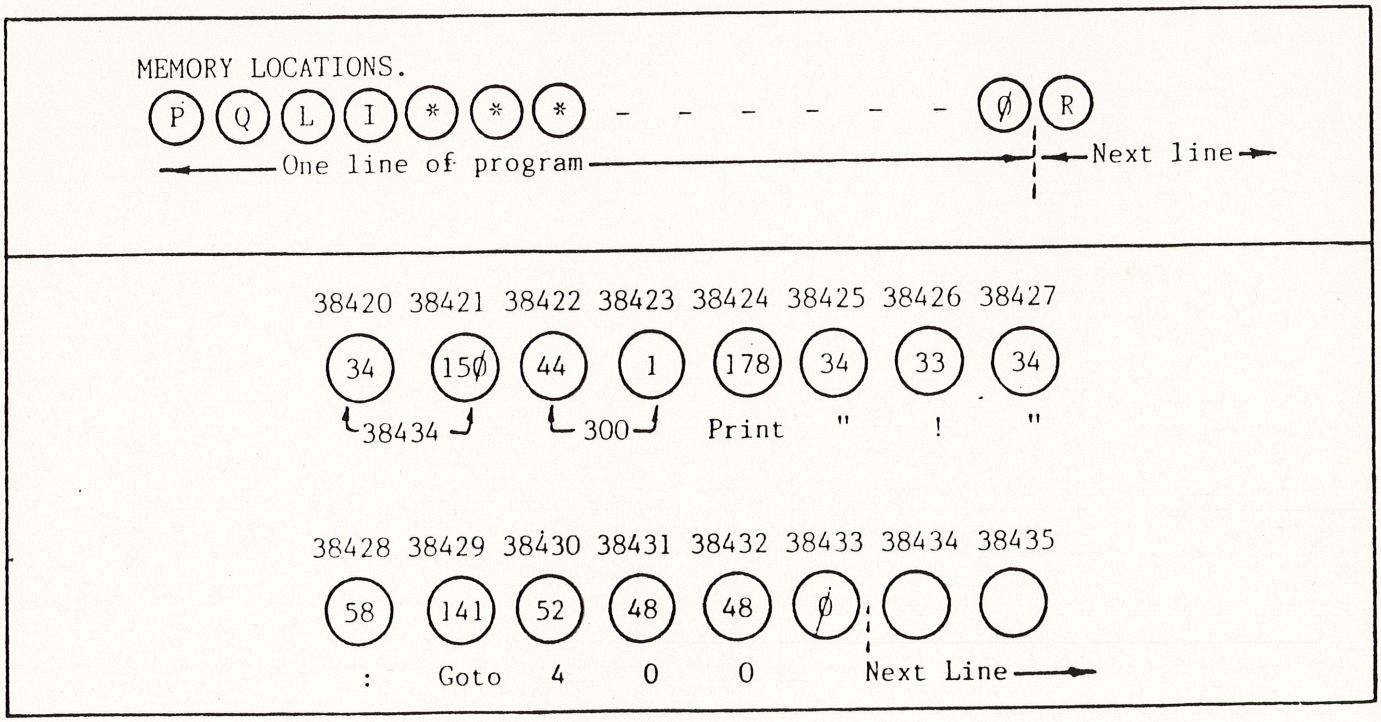
How to use this program:-

1) Type in and CSAVE as listed.
2) Before keying in your next program load the renumbering program from tape.

3) Key in your program with particular attention to the following.
   a) Line numbers used and called must be in the range 1-9999
   b) All line numbers or subroutines quoted within the text of a program line must be preceeded by GOTO or GOSUB and be right justified in a 4 space field. This means that 5 digit numbers if used will be seen only as the first four digits from the left and therefore will not be found as an existing number.

   i.e. IF ...THEN20ELSE325 must be entered as
   IF...THENGOTO 20ELSEGOTO 325
   (the line number 20 is preceeded by two spaces the number 325 by one, to create a 4 space field for the number - This allows say a two digit number to be reassigned as a three or four digit number)

   4 space field for the number - This allows say a two digit number to be reassigned as a three or four digit number)
   c) Line 10010:-
   Dimmension N%() greater than the number of line numbers in the section of program to be renumbered - A generous guess will do unless you are short of memory.
   :- Set the value of variable S to create a "safe zone" which the renumbering program will not alter. Normally this value will be 10000 (the first line number of the renumbering program itself) or it may be less if you wish to create a "gap" in line numbers between two sections of program - say between a main operating section and another section containing subroutines or Data lines. Remember that nothing in the "safe zone" is altered to a GOTO or GOSUB calling a lower section renumbered line would have to be changed separately.
4) Always CSAVE BEFORE running this program. If for any reason the renumbering is not totally successful then what remains of your program will probably be useless as part will be renumbered and part will not - equivalent to a population explosion of bags.
5) Key in RUN10000
   (If the result is a BAD SUBSCRIPT ERROR IN 10090 then increase the size of N%() in line 10010 - reloading will not be necessary as nothing has been altered yet.)
6) Enter 1st line number and increment on prompt.
7) When the renumbering is complete the cursor will return and the computer will be in READY mode (The time to execute



will be about 1½ seconds per program line)

8) Make any changes as indicated by messages printed during execution. (You can BREAK the execution to copy notes from the screen if it gets too full and then enter CONT to continue).
9) Probably a good idea to CSAVE again

10) Thoroughly test the renumbered section of the program - any problems - reload that saved at (4) and reRUN 10000 - it is not unknown for gremlins to be about for one renumbering run but absent for the next.

Program Operation

LINES 10000-10090

The line numbers are stored in array N%. Variable M is initialised to 31465 (BASIC Program start) and moved to the 1st byte of each program line by the calculation based on the values of P and Q the "start - position of next line" pointers. While M holds the decimal value of the memory positions the value of variable A is used in the PEEK'S. This is because PEEK and POKE will only work for the range -32768 A 32767 values for memory above 32767 must have 65536 subtracted before PEEK or POKE.
Line 10070 ends execution if P and Q both are zero - this occurs when the end-of-program byte sequence is found. (Two zero bytes then a 4 byte.

LINES 10100-10120

The first line number and increment are entered and edited so that they are both positive integers greater than zero. At 10120 a check is made as to whether the new numbering reach into the "safe zone" of line numbers.

LINE 10130

Reinitialise M back to program start and variable c to 1, C is the number of the line ie 1st 2nd 3rd etc.

LINES 10140-10180

Calculate new line number and POKE new line number to the appropriate bytes. M is now at the first byte of the line's storage sequence. The execution ends at 10160 if the "safe zone" is found.

LINES 10190-10210

Calculate R the memory position of the start of the next line.

LINES 10220-10290

This section searches through the program line contents looking at the value in each byte:

0: - end of line - go to next line
34: - Quote marks in a PRINT or INPUT statement - ignore
136: - DATA - ignore whole line
147: - REM - ignore whole line
141: - GOTO or 145 GOSUB: - alter line number called.

LINES 10300-10380

M set to the units column position of the number field area. The string variable 0$ is loaded with the characters of the line number field, and variable 0 is given the value of the

```
10000 '"Už-200  LINE RENUMBERING G.A MAR
SDEN 1984
10010 DIM N%(200):M=31465:C=1:S=10000'"F
IRST LINE OF SAFE ZONE"
10020 A=M:IFA>32767 THEN A=A-65536
10030 P=PEEK(A):A=M+1:IF A>32767 THEN A=
A-65536
10040 Q=PEEK(A):A=M+2:IFA>32767 THEN A=A
-65536
10050 L=PEEK(A):A=M+3:IF A>32767 THEN A=
A-65536
10060 I=PEEK(A)
10070 IF P=0ANDQ=0THEN PRINT "SAFE LINE
PAST END  ":END
10080 IFL+256*I>=STHEND=C-1:PRINTD;"LINE
S FOUND":GOTO10100
10090 N%(C)=L+256*I:M=P+256*Q:C=C+1:GOTO
10020
10100 INPUT"1ST LINE N0.";L$:F=INT(VAL(L
$)):IF F<1 THENGOTO10100
10110 INPUT"INCREMENT";I$:J=INT(VAL(I$))
:IFJ<1THENJ=1
10120 IF F+J*(C-1)>=STHEN PRINT"VALUES T
OO LARGE":GOTO10100
10130 M=31465:C=1
10140 A=M+2:IFA>32767THENA=A-65536
10150 B=M+3:IF B>32767A=A-65536
10160 IF PEEK(A)+256*PEEK(B)>=STHEN PRIN
T"FINISHED":END
10170 W=F+(C-1)*J
10180 POKEA,W-256*INT(W/256):POKEB,INT(W
/256)
10190 A=M:IFA>32767 THENA=A-65536
10200 B=M+1:IFB>32767 THENB=B-65536
10210 R=PEEK(A)+256*PEEK(B)
10220 M=M+4:T=1
10230 A=M:IF A>32767 THEN A=A-65536
10240 IF PEEK(A)=136OR PEEK(A)=147 THEN
M=R:C=C+1:GOTO10140
10250 IFPEEK(A)=145OR PEEK(A)=141 THEN G
OTO10300
10260 IF PEEK(A)=34THEN T=T*-1
10270 IF PEEK(A)=0 THEN M=R:C=C+1:GOTO10
140
10280 M=M+1
10290 IFT<0 THENGOTO10260ELSE GOTO10230
10300 M=M+4
10310 0$=""
10320 FORG=3TO0STEP-1
10330 A=M-G:IFA>32767THENA=A-65536
10340 0$=0$+CHR$(PEEK(A))
10350 NEXTG
```

2 of 3.

line number. Lastly a check to see if the line number called is within the safe zone.

LINES 10390-10470

The position of the old line no 0 is found in the arrayN%() H% is the position in the array that the value of 0 is compared with, in line 10470, and is initialised at the middle of the occupied area of the array. K% is initialised at just over¼ of the occupied length of the array. K% is reduced to just over ½ its value at each loop and H% is altered by adding or subtracing K% depending on which direction the search for the line number must go. The values of the last two array positions looked at are held in HL% and HP%. If a second look is taken at any array position then the conclusion is that the number does not exist and the error message at 10460 is printed and the search for another GOTO or GOSUB resumes at 10230. This search routine takes only 5 or 6 loops to find a number in an array of 70 line numbers and is therefore more efficient than just starting at the bottom and looking at each array position on the way up, which would take an average 35 loops to find a line number Of course it relies on the fact that the numbers are stored in numerical order. The tests at 10430 and 10440 are to see if the search has gone beyond the occupied range of the array and modify H% and K% accordingly. This routine is useful to look through any array of values providing they are in number order. (ascending or descending).

LINES 10480-10510

String variable NN$ is set to the characters of the new line number (including spaces) to be inserted in the 4 byte field of the program line. Line 10500 allows the start positions of subroutines to be recorded as the renumbering goes on. This line could be ommited and the start of subroutines marked in the program listing using REM" or "'. The " allows the remarks to be put in inverse characters to make them stand out as the program zooms up the screen after LIST.

LINES 10520-10620

This segment ensures that the value of 0 is consistant with the number of spaces at the left of the 4 byte field that the number came from.

LINES 10630-10670

Character by character POKEing of the new line number to its position in the line format.

```
10360 O=VAL(O$)
10370 IFO>=STHEN PRINT"LINE";W;"(NEW):-"
;OELSE GOTO 10430
10380 PRINT"[TWELVE SPACES]---INSIDE SAF
E ZONE:M=M+1:GOTO10230
10390 HPx=0:HLx=0
10400 Hx=1+D/2:Kx=D/2
10410 Kx(Kx+1)/2:HPx=HLx:HLx=Hx
10420 Hx=Hx+SGN(0-Nx(Hx))*Kx
10430 IFNx(Hx)=0THENHx=Hx-Kx:Kx=1
10440 IF Hx<1THENHx=1:Kx=1
10450 IFHx=HPxTHEN GOTO10460ELSEGOTO1047
0
10460 PRINT"LINE";W;"(NEW):- ";O;"NOT FO
UND":M=M+1:GOTO10230
10470 IFNx(Hx)<>0THENGOTO10410
10480 NN$="[3 SPACES]"+STR$(F+(Hx-1)*J)
10490 A=M-4:IFA>32767 THENA=A-65536
10500 IF PEEK(A)=145THEN PRINT"NEW SUBR@
 ";NN$"CALLED@";W
10510 NN$=RIGHT$(NN$,4)
10520 IFO>=1000THENGOTO10630
10530 A=M-3:IFA>32767THENA=A-65536
10540 IF PEEK(A)<>32THENGOTO10610
10550 IF O>=100THENGOTO10630
10560 A=M-2:IFA>32767THENA=A-65536
10570 IFPEEK(A)<>32THENGOTO10610
10580 IFO>=10THENGOTO10630
10590 A=M-1:IFA>32767THENA=A-65536
10600 IF PEEK(A)=32THENGOTO10630
10610 PRINT"LINE";W;"(NEW):FIELD ERROR";
O
10620 PRINT"[4 SPACES]---CHANGE TO NEW N
O.: ";NN$:GOTO10670
10630 FORG=1TO4
10640 A=M-4+G:IFA>32767THENA=A-65536
10650 POKEA,ASC(MID$(NN$,G,1))
10660 NEXTG
10670 M=M+1:GOTO10230
```